PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-318818
(43)Date of publication of application : 31.10.2002

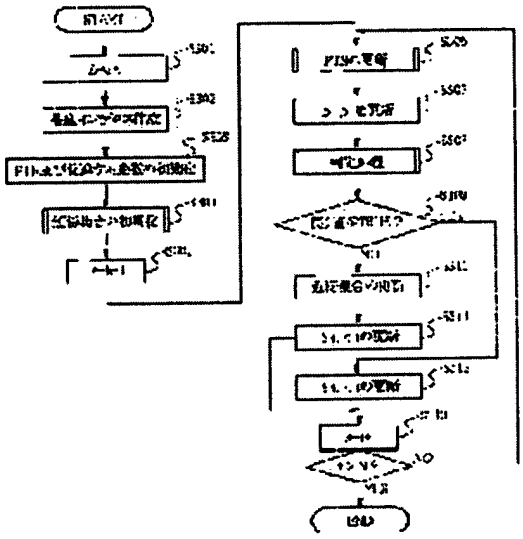
(51)Int.CI. G06F 17/30
G06F 17/16
G06T 7/00

(21)Application number : 2001-122755
(22)Date of filing : 20.04.2001

(71)Applicant : CANON INC
(72)Inventor : WASHISAWA TERUYOSHI

(54) DATA PROCESSING DEVICE AND ITS METHOD, AND ITS PROGRAM

(57)Abstract:
PROBLEM TO BE SOLVED: To make data retrieval executable at a high speed by similarity based on an inner product relative to given vector data.
SOLUTION: A sorting list relative to each component of a vector which is to be a prototype is made, and the nearest value to m-component values of test vectors X and -X are retrieved from the list, and respective positions are housed in PTR+ and PTR- (S303). A neighboring set to the number of k is initialized (S304), and a variable related to PTR is renewed (S306). Prototypes Yt are taken out successively, and determination processing is executed. If the similarity is larger than the minimum value up to the present, the prototype corresponding to the minimum value is replaced with Yt, and the neighboring set and the minimum value are renewed (S308-311).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] It is the data processor which extracts a prototype of a predetermined individual similar to the 1st given test vector from a prototype set described by multi-dimension vector as a retrieval result. A pretreatment means to create a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each component of said multi-dimension vector, and to calculate a square of a norm of each prototype, A test vector generation means to generate the 2nd test vector from said 1st test vector, A similarity count means to calculate similarity based on an absolute value of an inner product of the 1st test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, A data processor characterized by having an abbreviation control means which omits future processings, and an updating means to update a retrieval result based on a value of said similarity, by decision of abbreviation conditions based on a value of said similarity.
[Claim 2] A data processor according to claim 1 characterized by the above-mentioned test vector generation means generating said 1st test vector and a vector which a norm is equal and is reverse as said 2nd test vector.
[Claim 3] When a dimension of a vector space as which said prototype and 1st test vector are expressed is set to d, said similarity count means If similarity is first calculated to 1-dimensional subspace, said abbreviation control means judges that of abbreviation conditions based on similarity in the 1-dimensional subspace concerned and these abbreviation conditions are not satisfied It is the data processor according to claim 1 which controls said similarity count means to raise a dimension of subspace and to perform similarity count, and is characterized by said updating means updating a retrieval result when not satisfying abbreviation conditions in addition, even if a dimension of subspace reaches d.
[Claim 4] A data processor according to claim 1 with which said similarity count means is characterized by calculating similarity using the 2nd test vector and a thing which reduced a square of a difference of a prototype from a square of the 1st test vector, a thing which reduced a square of a difference of a prototype, and a norm of a prototype from a square of a norm of a prototype.
[Claim 5] A data processor according to claim 1 characterized by having a recognition means to recognize a class to which said 1st test vector belongs, based on a class to which a prototype of said predetermined individual extracted as a retrieval result belongs.
[Claim 6] It is the data-processing method of extracting a prototype of a predetermined individual similar to the 1st given test vector from a prototype set described by multi-dimension vector as a retrieval result. A head end process which creates a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each component of said multi-dimension vector, and calculates a square of a norm of each prototype, A test vector generation production process which generates the 2nd test vector from said 1st test vector, A similarity count production process which calculates similarity based on an absolute value of an inner product of the 1st test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, A data-processing method characterized by having an abbreviation control production process of omitting future processings, and an updating production process which updates a retrieval result based on a value of said similarity by decision of abbreviation conditions based on a value of said similarity.
[Claim 7] It is the data-processing program which extracts a prototype of a predetermined individual similar to the 1st given test vector from a prototype set described by multi-dimension vector as a retrieval result. A head end process which creates a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each component of said multi-dimension vector, and calculates a square of a norm of each prototype, A test vector generation production process which generates the 2nd test vector from said 1st test vector, A similarity count production process which calculates similarity based on an absolute value of an inner product of the 1st

test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, A data-processing program for making a computer perform an abbreviation control production process of omitting future processings, and an updating production process which updates a retrieval result based on a value of said similarity by decision of abbreviation conditions based on a value of said similarity.

[Translation done.]

* NOTICES *

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] This invention relates to the equipment which searches or recognizes data by the similarity based on the inner product between the data expressed especially by the vector about the data processor which performs retrieval or recognition of data.
[0002]
[Description of the Prior Art] The inner product is widely used as similarity between the data expressed by the vector. For example, in character recognition or a voice recognition system, the sampled data is mapped to the characteristic quantity space stretched at the suitable base, and the data by which vectorial representation was carried out is memorized as a prototype. The inner product of the data and the prototype which were newly inputted is calculated, and input data is identified as a thing belonging to the class corresponding to the nearest prototype. Most, the bad count method of effectiveness is exhaustive search, and the computational complexity becomes the order of the dimension of a vector, and the product of the number of prototypes.
[0003] it comes out in data base retrieval that the computational complexity of an inner product is recognized as a decisive failure. A data base can store now not only document data but an image and non-text data called voice by fast development of computer processing capacity in recent years. A keyword must be given in order to search non-[these] text data with the conventional keyword. If you want to avoid the time and effort of keyword grant, similar retrieval by the characteristic quantity vector must be performed.
[0004] Moreover, even when searching document data, in order to enable more flexible retrieval, a document is expressed by the vector, and the algorithm which performs similar retrieval based on this vector is also realized. At this time, said computational complexity poses an essential problem of retrieval system implementation. The data number of cases stored in the usual data base exceeds hundreds of thousands of affairs. Therefore, it becomes the hopeless condition that computational complexity increases hundreds of thousands of times whenever one number of dimension of a vector increases. In order to avoid such a condition, the number of data which reduces the dimension of a vector or is calculated must be reduced.
[0005] Since the former is equal to reducing the dimension of the presentation space of data, information required for data retrieval may not be enough expressed as a component of a vector. The latter is methodology which as for the number of data demanded as a retrieval result has a meaning when sufficiently small compared with the total of data. The problem dealt with by k-NN retrieval is this kind of thing, and some effective methods are proposed.
[0006] k-NN retrieval is a method for searching for k near a test vector and identifying the class of a test vector based on those classes from the set of the prototype memorized by the system. In such a case, it is one of the important technical problems how k prototypes near a test vector can be found at a high speed. Such a demand exists also in data base retrieval. A retrieval user wants only some data near the search key which he specified among a lot of data stored in the data base, and no value is found out to other data, furthermore the value of an inner product, etc. The technology for meeting such a demand of a retrieval user is in agreement with the purpose of the high-speed algorithm of k-NN retrieval.
[0007] When extracting k near a test vector from the set of a prototype, in order to mitigate the computational complexity of retrieval, it is common to structure the set of a prototype beforehand. Mitigation of the computational complexity of retrieval is expectable, so that it will carry out, if the property of data is made to reflect in the case of structuring. For example, in the case where a prototype is structured hierarchical, it is attained by repeating recursively actuation of dividing N dimension vector space where the prototype is expressed.
[0008] It is the method of dividing that whose boundary used when carrying out division management of the space is a

hyperplane in K-D-B Tree [reference 1] and a rectangle field R-Tree It is SS-Tree about the method of dividing by [reference 2] and hypersphere. It is SR-Tree about the method of dividing space in the combination of [reference 3], a rectangle, and hypersphere. It is called [reference 4]. If N dimension vector space is furthermore rechanged into the space stretched by the characteristic vector about the covariance matrix of distribution of a prototype, more effective structuring is expectable to the computational complexity of retrieval [reference 5 and 6].

[0009] However, the application to the data with which the computational complexity and storage capacity for data-structure-izing increase exponentially to the increment in the number of dimension of a vector, and these methods are expressed by the high order former vector may be restricted as a matter of fact. Furthermore, to a fatal thing, the similarity based on an inner product is incalculable, if a test vector is not given. Therefore, when an inner product defines similarity, structuring of such data cannot be used.

[0010] [reference 1] -- JT. Robinson: "The K-D-B Tree: A Search Structure forLarge Multidimensional Dynamic Indexes and" Proc. On ACM SIGMOD, pp.10-18 1981.[reference 2]A. Guttman: "R-trees: A dynamic index structure for spatial searching and" Proc. ACM SIGMOD, Boston USA pp.47-57, Jun.1984.[reference 3]DA. White andR.Jain: "Similarity indexing with the SS-tree," Proc. Of the 12th Int. Conf. On Data Engineering, NewOrleans USA pp.323-331 Feb.. [1996] [reference 4] Katayama, Sato: "Proposal of the index structure for the maximum contiguity retrieval to SR-Tree:high dimension point data" ****** (D-I), vol.18-D-I no.8 pp.703-717, Aug. 1997. [reference 5] RF.Sproull: "Refinemants to Nearest Neighbor Searchingin k-Dimensional Trees and" Algorithmica 6 pp.579-589 991.[reference 6] DA. Hite and R. Jain: "Similarity Indexing: Algorithmsand Performance," Proc. On SPIE pp.62-73 1996. [0011] On the other hand, there are also the loose structuring and the "loose" algorithm somewhat "wise" which has attained mitigation of computational complexity by the search algorithm which does not use a statistical property. This kind of algorithm is the only strategy which can be taken to the similar retrieval based on an inner product. Among those, one of the most fundamental things is the algorithm of Friedman and others called a projection algorithm [reference 7].

[Reference 7] Friedman F. Baskett and LJ. Shustek: "An Algorithm for Finding Nearest Neighbors" IEEE Structuring of the data demanded as pretreatment with JH. Trans. On Computers, pp.1000-1006, and a Oct. 1975. projection algorithm is sorting for every component of a vector, and is structuring based on a phase. That is, when a prototype is a d-dimensional vector, the sorting list of d pieces will be generated.

[0012] Two kinds, the list Vj which stored j component value put in order by ascending order by this processing, and the list Ij which stored the corresponding prototype ID number, are acquired only for the number of dimension of a vector. That is, the n+1st values Vj (n+1) are beyond the n-th values Vj (n) from the head of Vj. Moreover, j component value YIj (n) of the prototype YIj (n) which makes Ij (n) an ID number, and (j) are in agreement with Vj (n).

[0013] The principle of the projection algorithm for extracting 1 set of what is the closest to a test vector X from a prototype set is explained using drawing 13 . Retrieval is performed along with the sorting lists Vm and Im of one piece chosen on suitable criteria. This supports having chosen m shaft among drawing. Since the data number sorting was carried out [the data number] by the component value is stored, as for Im, the list on a list is reflecting the phase on m shaft correctly. The value nearest to the m component X of a test vector X (m) is first looked for from Vm. This is set to Vm (j). The prototype corresponding to this is YIm (j). By a diagram, YIm (j) is Y1. Although Y1 is the closest to X about m component, the distance in the whole space is not necessarily the nearest.

[0014] Now, next, the distance rho of X and Y1 (X, Y1) is calculated. Then, only the prototype with which the value of m component belongs to an open interval (X (m) rho (X, Y1), X(m)+rho (X, Y1)) (the open interval A in drawing) may be closer to X than Y1, and it turns out that it is meaningful as an object of retrieval. In the example of drawing, the set of the prototype used as the candidate for retrieval is restricted further (X (m) rho (X, Y2), X(m)+rho (X, Y2)) (the open interval B in drawing) by inspecting the prototype Y2 near a degree about m component. Thus, it is a projection algorithm that mitigation of computational complexity is in drawing by making small the prototype set set as the object of retrieval based on the component value in 1-dimensional subspace.

[0015]

[Problem(s) to be Solved by the Invention] Friedman's and others projection algorithm mentioned above is a method for the similar retrieval based on distance, and was not able to be applied for the similar retrieval based on the absolute value of an inner product, or a square.

[0016]

[Means for Solving the Problem] In order to solve a technical problem mentioned above, according to this invention, from a prototype set described by multi-dimension vector To a data processor extracted as a retrieval result, a prototype of a predetermined individual similar to the 1st given test vector A pretreatment means to create a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each

component of said multi-dimension vector, and to calculate a square of a norm of each prototype, A test vector generation means to generate the 2nd test vector from said 1st test vector, A similarity count means to calculate similarity based on an absolute value of an inner product of the 1st test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, By decision of abbreviation conditions based on a value of said similarity, it has an abbreviation control means which omits future processings, and an updating means to update a retrieval result based on a value of said similarity.

[0017] moreover, according to other modes of this invention, from a prototype set described by multi-dimension vector To a data-processing method extracted as a retrieval result, a prototype of a predetermined individual similar to the 1st given test vector A head end process which creates a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each component of said multi-dimension vector, and calculates a square of a norm of each prototype, A test vector generation production process which generates the 2nd test vector from said 1st test vector, A similarity count production process which calculates similarity based on an absolute value of an inner product of the 1st test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, By decision of abbreviation conditions based on a value of said similarity, it has an abbreviation control production process of omitting future processings, and an updating production process which updates a retrieval result based on a value of said similarity.

[0018] By furthermore, the data-processing program which extracts a prototype of a predetermined individual which according to other modes of this invention gives and is similar to the 1st test vector of ******** from a prototype set described by multi-dimension vector as a retrieval result A head end process which creates a list which arranged an identifier and a component value of each prototype of said prototype set in order of a component value about each component of said multi-dimension vector, and calculates a square of a norm of each prototype, A test vector generation production process which generates the 2nd test vector from said 1st test vector, A similarity count production process which calculates similarity based on an absolute value of an inner product of the 1st test vector and each prototype concerned using a square of said list and a norm of each of said prototype, and the 1st and 2nd test vectors, A computer is made to perform an abbreviation control production process of omitting future processings, and an updating production process which updates a retrieval result based on a value of said similarity by decision of abbreviation conditions based on a value of said similarity.

[0019]

[Embodiment of the Invention] With the <operation gestalt of ** 1st> book operation gestalt, the relational expression of an inner product and distance was drawn and the projection algorithm was used as high-speed calculus of an inner product. Furthermore, in order to evaluate an absolute value, the inner product to the given test vector and the inner product to a test vector and the vector of hard flow were evaluated. Therefore, it is the hard structuring and the "hard" general-purpose thing which does not need the parameter which should be set up beforehand to a prototype set.

[0020] Before giving details of the algorithm of this operation gestalt, the definition of a problem setup, a phrase, and a mark is clarified.

[0021] :omega= {Y1, Y2, ..., YN} and Yj**Rd which set the set of the prototype Yj of N individual with which the problem to assume was expressed as a d-dimensional vector to omega -- at this time, k large prototypes are extracted from omega about the absolute value of measuring rhoG (X, Yj) to test vector X** Rd given suitably. However, measuring rhoG (X, Yj) is : [0022] defined as an inner product.

[External Character 1]

$$\rho_G(X,Y_j) = X^T G Y_j = \sum_{n=1}^{n=d} \sum_{m=1}^{m=d} G(m,n)\, X(m)\, Y_j(n) \qquad (1)$$

Here, it wrote X (k) and Yj [a test vector X and k component value of Prototype Yj] (k), respectively.

[0023] the feature of this operation gestalt is the about the same as measuring rhoG (X, Yj) -- it is having adopted delta (Z, Yj) mentioned later as a function which gives a phase. It is equal to evaluating a prototype to descending of rhoG (X, Yj) and rhoG (- X, Yj) to extract a prototype to descending of the absolute value of rhoG (X, Yj). That is, if rhoG (- X, Yj) is large even if rhoG (X, Yj) is small, the absolute value of Yj of an inner product is large.

[0024] : which can divide rhoG (X, Yj) and rhoG (- X, Yj) into two steps of following processings -- Z=GXrhoG (X, Yj) =XTGY=(GX) TY=ZTY=rho (Z, Yj) rhoG(- X, Yj) =-XTGY=-(GX) TY=-ZTY=rho (- Z, Yj) -- rho (Z, Yj) and rho (- Z, Yj) are the inner products in orthonormal system here. On the other hand A ||2+||Yj||2-2rho (Z, Yj) top type is transformed further. :||Z-Yj||2=(Z-Yj)T(Z-Yj)=||Z which develops the square of the distance of Z and Yj and obtains a degree type -- : which obtains a degree type -- 2rho(Z, Yj)-||Z||2 -- : which defines the right-hand side (or left part) of a =||Yj||2-||Z-Yj||2 top type as a new function delta (X, Yj) -- now [delta(- Z, Yj) =||Yj||2-||Z+Yj||2] like delta(Z, Yj)

=||Yj||2-||Z-Yj||2 Suppose that the absolute value of k prototypes and an inner product with those test vectors X is given. The minimum value is set to xi among the absolute values of these inner products. These k pieces presuppose that it is a recently contact in this time.
[0025] Since Prototype Yj has the absolute value of an inner product smaller [the absolute value of an inner product with the newly given prototypes Yj and X is smaller than xi or] than all k prototypes if equal, or it is equal, it is guaranteed that it is not possible with k recently dots. That is, it can judge by whether a degree type is satisfied. : (xi>=delta (Z, Yj)) ** (xi>=delta (- Z, Yj)) A dot set can be updated (2), thus recently. On the other hand, :||Z-Yj||2 >=||P(Z-Yj) ||2 in which the following inequalities are materialized about the 2nd term of the right-hand side of the definition type of delta (Z, Yj), however P are the orthogonal projection operators to subspace. Therefore, delta(Z, Yj) <=||Yj||2-||P(Z-Yj) ||2 (3) Similarly delta(- Z, Yj) <=||Yj||2-||P(Z+Yj) ||2 (4) A criteria (2) type is :(xi>=||Yj||2-||P(Z-Yj) ||2) ** (xi>=||Yj||2-||P(Z+Yj) ||2) rewritten as follows using (3) and (4) types. (5) for example If the difference about m component of a vector, square ||Z(m)-Yj(m) ||2 of the sum, and ||Z(m)+Yj(m) ||2 are calculated and the criteria of (5) types are fulfilled, it turns out that Yj is not contained in a dot recently.
[0026] (5) Evaluation by the formula is effective in respect of computational complexity rather than it uses (2) types. If the computational complexity for evaluating (2) types is actually estimated, since it is calculable beforehand, it is not necessary to take into consideration the 1st term of the right-hand side. As for the 2nd term, the multiplication for a number of dimension of a vector and addition are needed. On the other hand, the multiplication for a number of dimension of subspace and addition are enough for (5) evaluation of expression. Therefore, the more the dimension of the subspace where (5) types are materialized is small, the more mitigation of computational complexity is expectable.
[0027] Hereafter, 1 operation gestalt of this invention is explained to details using a drawing.
[0028] [Operation gestalt 1] The configuration of the data retrieval equipment of this operation gestalt is shown in drawing 1 . I/O device 101 is equipped with a means to input retrieval data and commands, such as a communication line and a keyboard, and a means to output retrieval results, such as a communications control means and a display. For example, in the case of the computer of a stand-alone, the data which this I/O device 101 consisted of the keyboard and the indicating equipment, transmitted the input data inputted from the keyboard to inner product count equipment 102, and was transmitted from inner product count equipment 102 is outputted to an indicating equipment.
[0029] On the other hand, in the case of the communication terminal connected to the communication line, this I/O device consists of a communication controller, and the input data inputted through the communication line is transmitted to similarity count equipment 102, and it transmits the data received from similarity count equipment 102 to the address specified through the communication line. The set of the prototype of N individual expressed by the d-dimensional vector is stored in the data base 103, and it is accessed by similarity count equipment 102.
[0030] The procedure of the <similarity count equipment 102>, next similarity count equipment 102 is explained using the flow chart of drawing 2 .
[0031] First, it inspects whether there was any input from I/O device 101 at step S201, and if there is nothing, step S201 is repeated, and if it is, it will progress to step S202. At step S202, it inspects whether it is that to which input data updates the data of a data base, and if that is right, and that is not right, it will progress to step S203 to step S204. At step S203, pretreatment mentioned later is performed, and if it is completed, it will progress to step S201. At step S204, it inspects whether it is the thing of computation, and if this input data is so, and it is not so, it will progress to step S205 to step S201. At step S205, retrieval processing mentioned later is performed, and if it is completed, it will progress to step S201.
[0032] Pretreatment performed at step S203 using drawing 9 is explained.
[0033] With pretreatment, the sorting list about the square and each component value of a norm of the prototype belonging to a prototype set is created. Since the former only calculates and memorizes square ||Yj||2 of the norm of Prototype Yj, below, the latter processing is explained. A list of two kinds called the list Vj which stored j component value put in order by ascending order by this processing, and the list Ij which stored the corresponding prototype ID number is obtained only for the number of dimension of a vector.
[0034] n is set as 1 at step S801. At step S802, the pair of a n-component value and an ID number is created about each prototype of N individual. namely, -- : -- {(Y1 (n), n) -- (-- Y -- two -- (-- n --) -- n --) ...} (YN (n), N) -- a step -- S -- 803 -- the above-mentioned pair -- the ascending order of a n-component value -- rearranging .
[0035] {(YIn (1), (n), In (1)) -- (-- YIn (2), (n), and In (2)) -- :Vn= which considers the list of a component value, and the list of an ID number as Lists Vn and In separately ... (YIn (N), (n), In (N)) and below} {YIn (1), (n), YIn (2) and (n), ..., YIn (N) and (n)} (6) In= {In (1), In (2), ..., In (N)} At the (7) step S804, n is incremented, if it is over the dimension d of a vector, processing will be ended, otherwise, it progresses to step S802. The relation of the list of two kinds is as follows. The n+1st values Vj (n+1) are beyond the n-th values Vj (n) from the head of Vj. Moreover, j

component value YIj (n) of the prototype YIj (n) which makes Ij (n) an ID number, and (j) are in agreement with Vj (n).
[0036] The retrieval processing performed at step S205 using drawing 3 is explained shortly. As an input of retrieval processing, the number k of the prototype demanded as the vector X for retrieval (it is called a test vector below), and metric tensor G and a retrieval result is given.
[0037] The index list of the base of vector space is created at the :Z=GX step S302 which multiplies a test vector by metric tensor from the left at step S301, and acquires Vector Z. :lambda= which this is a list which defines the sequence of the base which applies the rejection conditions mentioned later, for example, is a list corresponding to descending of the absolute value of the component value of a test vector X {lambda1, lambda2, ..., lambdad} (8) -- : which writes L sets to be lambdaL from the one smaller again -- lambdaL= {lambdad-L +1, lambdad-L +2, ..., lambdad} (9) -- m is further set as lambda 1.
[0038] At step S303, initialization processing of the variable PTR(ed) and related is performed. This processing is explained using drawing 4 .
[0039] The sorting list Vm of m component values is acquired at step S401. Step S402 is searched for the value nearest to m component value X of a test vector (m) from Vm, and the location is stored in PTR+ at it. namely, : -- |Vm(PTR+) X(m) |<=|Vm(j) X(m) | and **j** {1, 2, ..., N} -- similarly, it searches for the value nearest to -X (m) from Vm, and the location is stored in PTR-. namely, : -- |Vm(PTR-)+ -- at the X(m) |<=|Vm(j)+X(m) | and **j** {1, 2, ..., N} step S403 : which initializes a related variable as follows PTR+L=PTR+ 1, BND+L=0, and CAL+L=0PTR+H=PTR++ k near sets are initialized at 1, BND+H=0, CAL+H=0 PTR-L=PTR -1, BND-L=0, CAL-L=0 PTR-H=PTR-, BND-H=0, and the CAL-H=0 PTR=PTR+ step S304. This processing is explained using drawing 5 .
[0040] At step S501, set N0 (X) will be initialized to empty class soon. At step S502, t is set as 1. At step S503, PTR later mentioned using drawing 6 is updated. Since the 1st term of the right-hand side of :second=Im(PTR) deltas=2|rho (Z, Yj) | which calculates the prototypes YIm (PTR) and delta s of ID number Im (PTR) at step S504, however a top type is calculated with pretreatment, only read-out from storage is required.
[0041] step S505 -- near set Nt- : which adds an ID number and the value of deltas to 1 (X) -- Nt(X) =Nt- 1 (X) + {(s, deltas) -- if t is incremented at the} step S506 and k is exceeded, it will progress to step S507, otherwise, will return to step S503.
[0042] At step S507, ID number s corresponding to xit-1 and it for the minimum value of deltas will be memorized as taut-1 in a set soon. Initialization of the near set to step S304 is completed above.
[0043] At step S305, t is set as k+1. Renewal of PTR is performed at step S306. This processing is explained using drawing 6 and 7.
[0044] At step S601, it is BND-L. 0 and PTR-L inspect whether it is 0, and if that is right, and that is not right, it will progress to step S602 to step S603. : which performs the following processings at step S602 -- if it inspects, it comes out so and it is [whether 0 and PTR-H have BND-H equal to PTR+H, and] at BND-L=1 and the Dx-L=infinity step S603 -- step S604 -- otherwise, it progresses to step S605. : which performs the following processings at step S604 -- if 0 and PTR+L inspect how [smaller than PTR-H] it is, and BND+L comes out so and there is at BND-H = 1 and the Dx-H=infinity step S605 -- step S606 -- otherwise, it progresses to step S607. : which performs the following processings at step S606 -- if 0 and PTR+H inspect whether they are two or more N, and BND+H comes out so and there is at BND-H=1, Dx-H=infinityBND+L=1, and the Dx+L=infinity step S607 -- step S608 -- otherwise, it progresses to step S609. : which performs the following processings at step S608 -- at BND+H=1 and the Dx+H=infinity step S609, the product of four numbers, BND-L, BND-H and BND+L, and BND+H, inspects whether it is 1, if it comes out so and is, retrieval processing will be ended, otherwise, it progresses to step S610.
[0045] At step S610, BND-L+CAL-L inspects whether it is 1, and if that is right, and that is not right, it will progress to step S611 to step S612. : which performs the following processings at step S611 -- if BND-H+CAL-H inspects whether it is 1, comes out so and there is at Dx-L=|Vm(PTR-L)-Z(m) |CAL-L=1 step S612 -- step S613 -- otherwise, it progresses to step S614. : which performs the following processings at step S613 -- if BND+L+CAL+L inspects whether it is 1, comes out so and there is at Dx-H=|Vm(PTR-H)-Z(m) |CAL-H=1 step S614 -- step S615 -- otherwise, it progresses to step S616. : which performs the following processings at step S615 -- if BND+H+CAL+H inspects whether it is 1, comes out so and there is at Dx+L=zVm(PTR+L)-Z(m) |CAL+L=1 step S616 -- step S617 -- otherwise, it progresses to step S618. : which performs the following processings at step S617 -- if Dx-L is smaller than any of Dx-H, Dx+L, and Dx+H at Dx+H=|Vm(PTR+H)-Z(m) |CAL+H=1 step S618 -- step S619 -- otherwise, it progresses to step S620.
[0046] At step S619, the following processings are performed, and in :Dx=Dx-L, PTR=PTR-L, and CAL-L=0 step S620 which end renewal of PTR of step S306, if Dx-H is smaller than any of Dx-L, Dx+L, and Dx+H, and that is not

right, it will progress to step S621 to step S622.
[0047] At step S621, the following processings are performed, and in :Dx=Dx-H, PTR=PTR-H, and CAL-H=0 step S622 which end renewal of PTR of step S306, if Dx+L is smaller than any of Dx-L, Dx-H, and Dx+H, and that is not right, it will progress to step S623 to step S624.
[0048] The following processings are performed at step S623, and at :Dx=Dx+L, PTR=PTR+L, and CAL+L=0 step S624 which end renewal of PTR of step S306, the following processings are performed, and by update process of :Dx=Dx+H and PTR=PTR+H which end renewal of PTR of step S306, and CAL+H=0PTR, if the value of the variable relevant to PTR is changed and conditions are satisfied, the retrieval processing itself shown in drawing 3 will be ended.
[0049] By step S307, a degree type estimates (5) types by judgment processing later mentioned per drawing 8 at the :rho+=||Ys||-|| Z||rho-=||Ys||-||Z|| step S308 which calculates rho+ and rho-.
[0050] At step S309, the return value of the processing performed at step S308 judges whether it is FALSE, and if it is FALSE, and that is not right, it will progress to step S310 to step S312.
[0051] At step S310, the element corresponding to xit-1 will be deleted from a set soon. : which adds the prototype under current processing -- Nt(X) <-Nt-1(X) {(taut-1, xit-1)}+ -- {(Im (PTR), Dx) -- at the} step S311, the minimum value of Dx of the elements of Nt (X) is stored in xit, the ID number is stored in taut, and it progresses to step S313.
[0052] At step S312, a degree type is performed, t is incremented at t-:xi t=xi t-1tau t=tau 1 step S313 which progresses to step S313, if the result exceeds N, processing will be ended, otherwise, it progresses to step S306.
[0053] The function count performed by judgment processing of step S308 is explained using drawing 8 .
[0054] At step S701, j is set as 1. A degree type is judged at 2Dx <-Dx-(Ys(lambdaj)-X (lambdaj)) 2 step S703. : which performs the following processings at step S702 -- rho+=rho+-(Ys(lambdaj)-X (lambdaj))2rho-=rho -- (Ys(lambdaj)+X (lambdaj)) -- If satisfied, TRUE will be set as a return value and processing will be ended, otherwise, it progresses to step S704. At step S704, if j is incremented and the result exceeds the dimension d of a vector, FALSE will be set as a return value, and processing will be ended, otherwise, it will return to step S702.
[0055] Nt (X) when ending at step S313 is outputted as a retrieval result.
[0056] The effect by the operation gestalt explained above is verified by computer simulation.
[0057] [computer simulation] -- the number of prototypes demanded as a retrieval result in order to verify the effectiveness of the operation gestalt mentioned above -- k= 10 pieces and several prototypes -- computer simulation of retrieval was conducted to N= 10000 pieces. dimension [of the : and the vector whose following items are experiment parameters]: -- d= -- {-- : and CPU:PentiumIII whose item of the computer used for the 10 unit} experiment from 10 to 100 is as follows, and - (500MHz) main memory:128 MB-OS:Linux -2.0.36 -- in addition, programming language used C.
[0058] [Experiment procedure]
(1) Generate the prototype set which consists of a d-dimensional vector N individual using a uniform random number.
(2) Use a uniform random number and it is d dimensions. One metric tensor of vector space is generated.
(3) Generate one d-dimensional test vector using a uniform random number.
(4) Perform exhaustive search.
(5) Perform retrieval by the proposal algorithm.
[0059] The average of the relative CPU time which shows the five above-mentioned procedures to a 10 times repeat and the following was calculated.
Relative CPU time = (CPU time of a proposal algorithm) a /(CPU time of exhaustive search) experimental result is shown in drawing 10 . The horizontal axis was shown for the dimension (number of dimensions) of a vector among drawing, and the axis of ordinate and the number N of prototypes were shown for the relative CPU time (CPU time ratio) as a parameter.
[0060] Drawing shows that it is not concerned with the number N of prototypes, but the relative CPU time is increasing to the primary order with the increment in the dimension of a vector. However, the inclination at the time of N= 10000 pieces is very small. The value of the relative CPU time at this time indicates the sufficiently small value to be-dimensional [100] or 18% also by high order origin 2% by ten dimensions.
[0061] <Operation gestalt 2> The case where the similarity count mentioned above is applied to recognition equipment is explained.
[0062] Drawing 11 is drawing showing the configuration of the recognition equipment of this operation gestalt. The set of the prototype of N individual expressed by the d-dimensional vector is stored in the data base 103, and two or more prototypes exist to each category used as the candidate for recognition, respectively. Others are the same as that of drawing 1 .

[0063] In for example, face recognition, the d-dimensional vector stored in a data base 103 can constitute by connecting the person's information with the image in two or more angles of an everybody object (Mr. A, Mr. B, ...), expression, etc. In such a case, by processing described below, an input image and k near images are chosen, the frequency of the person number connected with each can be totaled, and the highest person of frequency can be specified as he is a person corresponding to an image.
[0064] Moreover, the procedure of similarity count equipment 102 follows drawing 2 like the operation gestalt 1. However, in step S204-205, retrieval processing turns into recognition processing which used retrieval. This recognition processing is explained using the flow chart of drawing 12 .
[0065] In drawing 12 , step S301-313 are the same as that of drawing 3 . At step S314, the frequency of the class number related with the ID number which will belong to a set soon is calculated. An ID number is rearranged into descending of the frequency of a class number at step S315. Thereby, a class candidate is obtained by order with a high possibility that a test vector belongs. Among these, a class with the largest frequency may be outputted as a recognition result, and a predetermined individual or all classes may be outputted as a recognition candidate in order of frequency.
[0066] In addition, even if it applies this invention to the equipment which consists of a single device, it may be applied to the system which consists of two or more devices. Moreover, the storage which memorized the program code of the software which realizes the function of the operation gestalt mentioned above may be supplied to equipment or a system, and you may attain by reading and performing the program code with which the computer in equipment or a system was stored in the storage.
[0067] Furthermore, by reading and performing the program code with which the computer in equipment or a system was stored in the storage, also when an above-mentioned function is realized, it is contained by processing of OS which it not only realizes directly the function of the operation gestalt mentioned above, but is working on a computer based on directions of the program code.
[0068] The storage which memorized the program code in these cases will constitute this invention.
[0069]
[Effect of the Invention] Since the similarity based on the absolute value of the inner product of a vector is calculable at a high speed according to this invention as explained above, based on this similarity, it is effective in the ability to perform retrieval of similar data and recognition of a class which belongs at a high speed to the given data by which vectorial representation was carried out.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－318818
(P2002－318818A)
(43)公開日　平成14年10月31日(2002.10.31)

(51)Int.Cl.'	識別記号	FI	テーマコード'(参考)
G06F 17/30	350	G06F 17/30	350C　5B056
	210		210D　5B075
17/16		17/16	K　5L096
G06T 7/00	300	G06T 7/00	300C
			300F

審査請求　未請求　請求項の数7　ＯＬ　（全 11 頁）

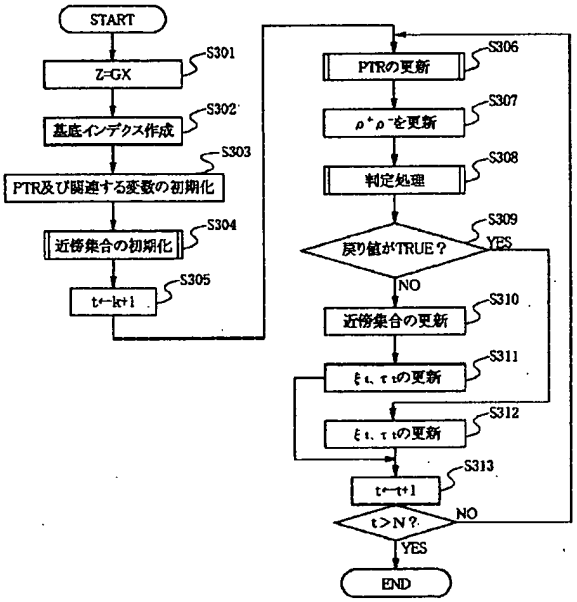
(21)出願番号　特願2001－122755(P2001－122755)
(22)出願日　平成13年４月20日(2001.4.20)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号
(72)発明者　鷲澤　輝芳
東京都大田区下丸子３丁目30番２号キヤノン株式会社内
(74)代理人　100090538
弁理士　西山　恵三　（外１名）

Ｆターム(参考)　5B056 BB42 HH00 HH03
5B075 NK06 NR12 PR06 QM08 QS20
5L096 JA03 JA26

(54)【発明の名称】　データ処理装置及びその方法、及びそのプログラム

(57)【要約】
【課題】　与えられたベクトルデータに対して内積を基にした類似度によるデータ検索が高速に実行できるようにする。
【解決手段】　プロトタイプとなるベクトルの各成分に対するソーティング・リストを作成しておき、テストベクトルX及び－Xのm成分値に最も近い値をリストから探索し、それぞれの位置をPTR⁺及びPTR⁻に格納する(S303)。k個の近傍集合の初期化を行い(S304)、PTRと関連する変数を更新する(S306)。順次プロトタイプY_tを取り出し、判定処理によって、類似度が今までの最小値より大きければ、最小値に対応するプロトタイプとY_tを入れ替え、近傍集合及び最小値を更新する(S308～311)。



【特許請求の範囲】

【請求項１】　多次元ベクトルで記述されたプロトタイプ集合から、与えられた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理装置であって、

前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理手段と、

前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成手段と、

前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算手段と、

前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御手段と、

前記類似度の値に基づいて検索結果を更新する更新手段とを有することを特徴とするデータ処理装置。

【請求項２】　上記テストベクトル生成手段が、前記第２のテストベクトルとして、前記第１のテストベクトルとノルムが等しく方向が逆であるようなベクトルを生成することを特徴とする請求項１に記載のデータ処理装置。

【請求項３】　前記プロトタイプ及び第１のテストベクトルが表現されているベクトル空間の次元をｄとすると、前記類似度計算手段は、まず１次元部分空間に対して類似度を計算し、

前記省略制御手段が、当該１次元部分空間における類似度に基づく省略条件のを判断し、該省略条件を満足しなければ、部分空間の次元を上げて類似度計算を行うように前記類似度計算手段を制御し、

前記更新手段は、部分空間の次元がｄに達してもなお省略条件を満足しないときに、検索結果を更新することを特徴とする請求項１に記載のデータ処理装置。

【請求項４】　前記類似度計算手段が、プロトタイプのノルムの２乗から第１のテストベクトルとプロトタイプの差の２乗を減じたものと、プロトタイプのノルムの２乗から第２のテストベクトルとプロトタイプの差の２乗を減じたものとを用いて、類似度を計算することを特徴とする請求項１に記載のデータ処理装置。

【請求項５】　検索結果として抽出された前記所定個のプロトタイプが属するクラスに基づいて、前記第１のテストベクトルの属するクラスを認識する認識手段を有することを特徴とする請求項１に記載のデータ処理装置。

【請求項６】　多次元ベクトルで記述されたプロトタイプ集合から、与えられた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理方法であって、

前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理工程と、

前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成工程と、

前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算工程と、

前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御工程と、

前記類似度の値に基づいて検索結果を更新する更新工程とを有することを特徴とするデータ処理方法。

【請求項７】　多次元ベクトルで記述されたプロトタイプ集合から、与えられた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理プログラムであって、

前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理工程と、

前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成工程と、

前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算工程と、

前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御工程と、

前記類似度の値に基づいて検索結果を更新する更新工程とをコンピュータに実行させるためのデータ処理プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、データの検索または認識を行なうデータ処理装置に関し、特にベクトルで表現されたデータ間の内積を基にした類似度によってデータを検索もしくは認識する装置に関するものである。

【０００２】

【従来の技術】ベクトルで表現されたデータ間の類似度として内積が広く用いられている。例えば文字認識や音声認識システムでは、サンプリングされたデータを適当な基底で張られた特徴量空間に写像し、ベクトル表現されたデータをプロトタイプとして記憶しておく。新たに入力されたデータとプロトタイプとの内積を計算し、入力データを最も近いプロトタイプに対応するクラスに属するものとして同定する。最も効率の悪い計算方法は全数探索であり、その計算量はベクトルの次元とプロトタイプ数の積のオーダーになる。

【０００３】内積の計算量が決定的な障害として認識さ

れるのは、データベース検索においてである。近年のコンピュータ処理能力の飛躍的発達によってデータベースは文書データのみならず、画像や音声といった非テクストデータをも蓄積できるようになった。これら非テクストデータを従来のキーワードで検索するためにはキーワードを付与しなければならない。キーワード付与の手間を避けたいなら特徴量ベクトルによる類似検索を行わなければならない。

【０００４】また文書データを検索する場合でも、より柔軟な検索を可能にするために文書をベクトルで表現し、このベクトルに基づく類似検索を行うアルゴリズムも実現されている。このとき前記計算量が検索システム実現の本質的問題となる。通常のデータベースに格納されるデータ件数は数十万件を超える。従ってベクトルの次元数が１個増加する度に計算量が数十万増加するという絶望的な状況になる。このような状況を回避するためには、ベクトルの次元を減らすか、計算するデータ数を減らすしかない。

【０００５】前者はデータの表現空間の次元を減らすことに等しいので、データ検索に必要な情報がベクトルの成分として十分表現されない可能性がある。後者は、検索結果として要求されているデータ数がデータの総数に比べて十分小さいときに意味のある方法論である。k-NN探索で取り扱う問題はこの種のものであり、いくつかの有効な方法が提案されている。

【０００６】k-NN探索は、システムに記憶されているプロトタイプの集合から、テストベクトルに近いk個を探索し、それらのクラスを基に、テストベクトルのクラスを同定するための方法である。このような場合、テストベクトルに近いk個のプロトタイプを如何に高速に見つけることができるかが重要な課題の１つである。このような要求はデータベース検索においても存在する。検索ユーザーは、データベースに格納されている大量のデータのうち、自分が指定した検索キーに近いいくつかのデータだけが欲しいのであって、他のデータの、ましてや内積の値などに何の価値も見出さない。検索ユーザーのこのような要求に応えるための技術は、k-NN探索の高速アルゴリズムの目的と一致する。

【０００７】プロトタイプの集合からテストベクトルに近いk個を抽出する場合、探索の計算量を軽減するために予めプロトタイプの集合を構造化しておくのが一般的である。構造化の際にデータの性質を反映させればさせるほど、探索の計算量の軽減が期待できる。例えばプロトタイプを階層的に構造化する場合では、プロトタイプが表現されているN次元ベクトル空間を分割するという操作を再帰的に繰り返すことによって達成される。

【０００８】空間を分割管理するときに用いられる境界が超平面であるものをK-D-B Tree〔文献１〕、矩形領域で分割する方法をR-Tree〔文献２〕、超球で分割する方法をSS-Tree〔文献３〕、矩形と超球との組み合わせで空間を分割する方法をSR-Tree〔文献４〕という。更にN次元ベクトル空間をプロトタイプの分布の共分散行列に関する固有ベクトルで張られる空間に変換し直しておけば、探索の計算量に対して、より効果的な構造化が期待できる〔文献５，６〕。

【０００９】しかしこれらの方法はデータ構造化のための計算量と記憶容量がベクトルの次元数の増加に対して指数関数的に増大してしまい、高次元ベクトルで表現されているデータへの応用が事実上制限されてしまう可能性がある。更に、致命的なことに、内積に基づく類似度は、テストベクトルが与えられなければ計算できない。従って、類似度を内積で定義した場合、このようなデータの構造化が利用できない。

【００１０】〔文献１〕JT. Robinson: "The K-D-B Tree: A Search Structure forLarge Multidimensional Dynamic Indexes," Proc. On ACM SIGMOD,pp.10-18, 1981.

〔文献２〕A. Guttman: "R-trees: A dynamic index structure for spatial searching," Proc. ACM SIGMOD, Boston, USA, pp.47-57, Jun.1984.

〔文献３〕DA. White and R. Jain: "Similarity indexing with the SS-tree," Proc. Of the 12th Int. Conf. On Data Engineering, NewOrleans, USA, pp.323-331, Feb. 1996.

〔文献４〕片山，佐藤："SR-Tree：高次元点データに対する最近接検索のためのインデクス構造の提案，"信学論(D-I), vol.18-D-I, no.8, pp.703-717,Aug. 1997.

〔文献５〕RF. Sproull: "Refinemants to Nearest Neighbor Searchingin k-Dimensional Trees," Algorithmica, 6, pp.579-589, 991.

〔文献６〕DA. Hite and R. Jain: "Similarity Indexing: Algorithmsand Performance," Proc. On SPIE, pp.62-73, 1996.

【００１１】一方、統計的性質を利用しない"緩い"構造化と、少し"賢い"探索アルゴリズムによって計算量の軽減を達成しているアルゴリズムもある。この種のアルゴリズムが、内積に基づく類似検索に対して取り得る唯一の戦略である。そのうち最も基本的なもののひとつが射影アルゴリズムと呼ばれるFriedmanらのアルゴリズムである〔文献７〕。

〔文献７〕JH. Friedman, F. Baskett, and LJ. Shustek: "An Algorithm for Finding Nearest Neighbors," IEEE Trans. On Computers,pp.1000-1006, Oct. 1975.

射影アルゴリズムで前処理として要求されるデータの構造化は、ベクトルの各成分毎のソーティングであり、位相に基づく構造化である。つまり、プロトタイプがd次元ベクトルのときは、d個のソーティング・リストが生成されることになる。

【００１２】この処理で、昇順に並べられたj成分値を格納したリストV_jと、対応するプロトタイプID番号を格納したリストI_jの２種類が、ベクトルの次元数だけ得られる。即ち、V_jの先頭からn+1番目の値$V_j(n+1)$はn番目の値$V_j(n)$以上である。また、$I_j(n)$をID番号とするプロトタイプ$Y_{I_j(n)}$のj成分値$Y_{I_j(n)}(j)$が$V_j(n)$と一致する。

【００１３】プロトタイプ集合からテストベクトルXに最も近いもの1組を抽出するための射影アルゴリズムの原理を図１３を用いて説明する。探索は、適当な基準で選択された１個のソーティング・リストV_mとI_mに沿って行われる。これは、図中、m軸を選択したことに対応している。I_mは成分値によってソーティングされたデータ番号が格納されているので、リスト上での並びがm軸上での位相を正確に反映している。まずテストベクトルXのm成分X(m)に最も近い値をV_mから探す。これを$V_m(j)$とする。これに対応するプロトタイプは$Y_{I_m(j)}$である。図では$Y_{I_m(j)}$がY_1である。Y_1はm成分に関してXに最も近いが、全空間での距離が最も近いとは限らない。

【００１４】さて次に、XとY_1との距離$\rho(X,Y_1)$を計算する。すると、m成分の値が開区間$(X(m)\rho(X,Y_1),X(m)+\rho(X,Y_1))$(図中の開区間Ａ）に属するプロトタイプのみが、$Y_1$よりXに近い可能性があり、探索の対象として意味があることがわかる。図の例では、m成分に関して次に近いプロトタイプY_2を検査することによって、探索対象となるプロトタイプの集合が更に$(X(m)\rho(X,Y_2),X(m)+\rho(X,Y_2))$(図中の開区間Ｂ）に制限される。このように1次元部分空間での成分値をもとに、探索の対象となるプロトタイプ集合を小さくしていくことによって、計算量の軽減を図っているのが射影アルゴリズムである。

【００１５】

【発明が解決しようとする課題】上述したFriedmanらの射影アルゴリズムは、距離に基づく類似検索のための方法であって、内積の絶対値、あるいは二乗に基づく類似検索のためには適用できなかった。

【００１６】

【課題を解決するための手段】上述した課題を解決するために、本発明によれば、多次元ベクトルで記述されたプロトタイプ集合から、与えられた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理装置に、前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理手段と、前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成手段と、前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算手段と、前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御手段と、前記類似度の値に基づいて検索結果を更新する更新手段とを備える。

【００１７】また、本発明の他の態様によれば、多次元ベクトルで記述されたプロトタイプ集合から、与えられた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理方法に、前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理工程と、前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成工程と、前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算工程と、前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御工程と、前記類似度の値に基づいて検索結果を更新する更新工程とを備える。

【００１８】更に、本発明の他の態様によれば、多次元ベクトルで記述されたプロトタイプ集合から、与えら得れた第１のテストベクトルに類似する所定個のプロトタイプを検索結果として抽出するデータ処理プログラムにより、前記多次元ベクトルの各成分につき、前記プロトタイプ集合の各プロトタイプの識別子と成分値とを成分値の順に並べたリストを作成し、各プロトタイプのノルムの２乗を計算する前処理工程と、前記第１のテストベクトルから第２のテストベクトルを生成するテストベクトル生成工程と、前記リスト及び前記各プロトタイプのノルムの２乗と、第１及び第２のテストベクトルとを用いて、当該第１のテストベクトルと各プロトタイプとの内積の絶対値に基づく類似度を計算する類似度計算工程と、前記類似度の値に基づく省略条件の判断により、以後の処理を省略する省略制御工程と、前記類似度の値に基づいて検索結果を更新する更新工程とをコンピュータに実行させる。

【００１９】

【発明の実施の形態】＜第１の実施形態＞本実施形態では、内積と距離との関係式を導出し、内積の高速計算法として、射影アルゴリズムを利用した。更に、絶対値を評価するために、与えられたテストベクトルに対する内積と、テストベクトルと逆方向のベクトルに対する内積とを評価した。従って、プロトタイプ集合に対する”硬い”構造化や、予め設定すべきパラメタを必要としない汎用的なものである。

【００２０】本実施形態のアルゴリズムの詳細を述べる前に、問題設定、語句と記号の定義を明らかにしておく。

【００２１】想定する問題は、d次元ベクトルとして表現されたN個のプロトタイプY_iの集合をΩとする：

$\Omega = \{Y_1, Y_2, \cdots, Y_N\}, Y_i \in R^d$

このとき、適当に与えられたテストベクトル$X \in R^d$に、計量$\rho_G(X,Y_i)$の絶対値に関して大きいk個のプロトタイプをΩから抽出する。ただし計量$\rho_G(X,Y_i)$は内積とし＊て定義される：

【００２２】

【外１】

$$\rho_G(X,Y_j) = X^T G Y_j = \sum_{n=1}^{n=d} \sum_{m=1}^{m=d} G(m,n)\, X(m)\, Y_j(n) \qquad (1)$$

ここで、テストベクトルX、プロトタイプY_iのk成分値をそれぞれX(k)、$Y_i(k)$と表記した。

【００２３】本実施形態の特徴は、計量$\rho_G(X,Y_j)$と同じ位相を与える関数として、後述する$\delta(Z,Y_i)$を採用したことである。$\rho_G(X,Y_i)$の絶対値の大きい順にプロトタイプを抽出するということは、$\rho_G(X,Y_i)$と$\rho_G(-X,Y_i)$の大きい順にプロトタイプを評価することに等しい。即ち、$\rho_G(X,Y_i)$が小さくても$\rho_G(-X,Y_i)$が大きければ、Y_iは内積の絶対値が大きい。

【００２４】$\rho_G(X,Y_i)$及び$\rho_G(-X,Y_i)$は以下のような2段階の処理に分割できる：

$Z = GX$

$\rho_G(X,Y_i) = X^T GY = (GX)^T Y = Z^T Y = \rho(Z,Y_i)$

$\rho_G(-X,Y_i) = -X^T GY = -(GX)^T Y = -Z^T Y = \rho(-Z,Y_i)$

ここで$\rho(Z,Y_i)$及び$\rho(-Z,Y_i)$は正規直交系における内積である。一方、ZとY$_i$との距離の２乗を展開して次式を得る：

$\| Z-Y_i \|^2 = (Z-Y_i)^T (Z-Y_i) = \| Z \|^2 + \| Y_i \|^2 - 2\rho(Z,Y_i)$※

このようにして、最近傍点集合を更新することが出来る。一方、$\delta(Z,Y_i)$の定義式の右辺第２項に関して以下の不等式が成立する：

$\| Z-Y_i \|^2 \geqq \| P(Z-Y_i) \|^2$

ただし、Pは部分空間への直交射影作用素である。従って

例えば、ベクトルのm成分に関する差と和の二乗$\| Z(m)-Y_i(m) \|^2$と$\| Z(m)+Y_i(m) \|^2$を計算して、(5)式の判定条件を満たせば、Y_iが最近傍点に含まれないことがわかる。

【００２６】(5)式での評価は、(2)式を利用するよりも、計算量の点で効果的である。実際、(2)式を評価するための計算量を概算してみると、右辺第１項は、予め計算できるから、考慮する必要はない。第２項は、ベクトルの次元数分の乗算と加算が必要になる。一方(5)式の評価には、部分空間の次元数分の乗算と加算で十分である。よって、(5)式が成立する部分空間の次元が小さければ小さいほど、計算量の軽減が期待出来る。

【００２７】以下、図面を用いて本発明の１実施形態を詳細に説明する。

【００２８】〔実施形態１〕図１に本実施形態のデータ検索装置の構成を示す。入出力装置１０１は、通信回線やキーボード等の検索データやコマンドを入力する手段

※上式を更に変形して次式を得る：

$2\rho(Z,Y_i) - \| Z \|^2 = \| Y_i \|^2 - \| Z-Y_i \|^2$

上式の右辺(或いは左辺)を新たな関数$\delta(X,Y_i)$として定義する：

$\delta(Z,Y_i) = \| Y_i \|^2 - \| Z-Y_i \|^2$

同様に

$\delta(-Z,Y_i) = \| Y_i \|^2 - \| Z+Y_i \|^2$

今、k個のプロトタイプと、それらのテストベクトルXとの内積の絶対値が与えられているとする。これら内積の絶対値のうち、最小値をξとする。このk個が、現時点での最近接点であるとする。

【００２５】新たに与えられたプロトタイプY_iとXとの内積の絶対値がξよりも小さいか等しければ、プロトタイプY_iは、k個の全てのプロトタイプよりも、内積の絶対値が小さいか等しいので、k個の最近傍点で有り得ないことが保証される。即ち、次式を満足するかどうかで判定できる：

$$(\xi \geqq \delta(Z,Y_i)) \wedge (\xi \geqq \delta(-Z,Y_i)) \qquad (2)$$

★ $\delta(Z,Y_i) \leqq \| Y_i \|^2 - \| P(Z-Y_i) \|^2 \qquad (3)$

同様に

$\delta(-Z,Y_i) \leqq \| Y_i \|^2 - \| P(Z+Y_i) \|^2 \qquad (4)$

判定条件(2)式は、(3)及び(4)式を用いて以下のように書き換えられる：

★

$$(\xi \geqq \| Y_i \|^2 - \| P(Z-Y_i) \|^2) \wedge (\xi \geqq \| Y_i \|^2 - \| P(Z+Y_i) \|^2) \qquad (5)$$

と、通信制御手段や表示装置等の検索結果を出力する手段を備える。例えばスタンドアロンのコンピュータの場合は、該入出力装置１０１はキーボードと表示装置から成り、キーボードから入力された入力データを内積計算装置１０２へ送信し、内積計算装置１０２から送信されたデータを表示装置に出力する。

【００２９】一方、通信回線に接続されている通信端末の場合は、該入出力装置は通信制御装置から成り、通信回線を介して入力された入力データを類似度計算装置１０２に送信し、類似度計算装置１０２から受信したデータを通信回線を介して指定されたアドレスに送信する。データベース１０３には、d次元ベクトルで表現されたN個のプロトタイプの集合が格納されており、類似度計算装置１０２によってアクセスされる。

【００３０】＜類似度計算装置１０２＞次に、類似度計算装置１０２の処理手順を図２のフローチャートを用いて説明する。

【００３１】まず、ステップS201で入出力装置１０１から入力があったかどうかを検査し、なければステップS201を繰り返し、あればステップS202へ進む。ステップS202では、入力データがデータベースのデータを更新するものかどうかを検査し、そうであればステップS203へ、そうでなければステップS204へ進む。ステップS203では、後述する前処理を実行し、それが終了したらステップS201へ進む。ステップS204では該入力データが計算処理のものかどうかを検査し、そうであればステップS205へ、そうでなければステップS201へ進む。ステップS205では、後述する検索処理を実行し、それが終了したらステップS201へ進む。

【００３２】図９を用いてステップS203で実行される前処理について説明する。

【００３３】前処理では、プロトタイプ集合に属するプロトタイプのノルムの２乗と、各成分値に関するソーティング・リストを作成する。前者はプロトタイプY_iのノルムの２乗$\|Y_i\|^2$を計算して記憶しておくだけなので、以下では後者の処理を説明する。この処理で、昇順に並べられたj成分値を格納したリストV_iと、対応するプロトタイプID番号を格納したリストI_iという２種類のリストが、ベクトルの次元数だけ得られる。

【００３４】ステップS801でnを１に設定する。ステップS802で、N個のプロトタイプそれぞれについて、n成分値とID番号のペアを作成する。即ち：

$\{(Y_1(n),n),(Y_2(n),n),\cdots,(Y_N(n),N)\}$

ステップS803で上記ペアをn成分値の昇順に並べ替える。

【００３５】$\{(YIn(1)(n),In(1)),(YIn(2)(n),In(2)),\cdots,(YIn(N)(n),In(N))\}$

以下、成分値の並びとID番号の並びを別々にリストVn、Inとする：

$$V_n=\{Y_{In(1)}(n),Y_{In(2)}(n),\cdots,Y_{In(N)}(n)\} \quad (6)$$

$$I_n=\{I_n(1),I_n(2),\cdots,I_n(N)\} \quad (7)$$

ステップS804ではnをインクレメントし、ベクトルの次元dを超えていたら処理を終了し、そうでなければステップS802へ進む。2種類のリストの関係は次のとおりである。V_iの先頭からn+1番目の値$V_i(n+1)$はn番目の値$V_i(n)$以上である。また、$I_i(n)$をID番号とするプロトタイプ$YI_i(n)$のi成分値$YI_i(n)(j)$が$V_i(n)$と一致する。

【００３６】今度は図３を用いてステップS205で実行される検索処理を説明する。検索処理の入力として、検索のためのベクトルX（以下テストベクトルと呼ぶ）と、計量テンソルG、検索結果として要求されているプロトタイプの個数kが与えられている。

【００３７】ステップS301ではテストベクトルに計量テンソルを左から乗じてベクトルZを得る：

$Z=GX$

ステップS302ではベクトル空間の基底のインデクスリストを作成する。これは後述する棄却条件を適用する基底の順序を定めるリストであり、例えば、テストベクトルXの成分値の絶対値の大きい順に対応するリストである：

$$\Lambda=\{\lambda_1,\lambda_2,\cdots,\lambda_d\} \quad (8)$$

また小さいほうからL個の集合をΛ_Lと書く：

$$\Lambda_L=\{\lambda_{d-L+1},\lambda_{d-L+2},\cdots,\lambda_d\} \quad (9)$$

更にmを$\lambda 1$に設定する。

【００３８】ステップS303ではPTR及び関連する変数の初期化処理を実行する。この処理を図４を用いて説明する。

【００３９】ステップS401でm成分値のソーティング・リストV_mを取得する。ステップS402ではテストベクトルのm成分値X(m)に最も近い値をV_mから探索し、その位置をPTR^+に格納する。即ち：

$|V_m(PTR^+)X(m)|\leqq|V_m(j)X(m)|,\forall_i\in\{1,2,\cdots,N\}$

同様に、-X(m)に最も近い値をV_mから探索し、その位置をPTR^-に格納する。即ち：

$|V_m(PTR^-)+X(m)|\leqq|Vm(j)+X(m)|,\forall j\in\{1,2,\cdots,N\}$

ステップS403では、関連する変数は以下のように初期化する：

$PTR^+_L=PTR^+_1,BND^+_L=0,CAL^+_L=0$

$PTR^+_H=PTR^+_1,BND^+_H=0,CAL^+_H=0$

$PTR^-_L=PTR^-_1,BND^-_L=0,CAL^-_L=0$

$PTR^-_H=PTR^-,BND^-_H=0,CAL^-_H=0$

$PTR=PTR^+$

ステップS304ではk個の近傍集合の初期化を行う。この処理を図５を用いて説明する。

【００４０】ステップS501では近傍集合$N_0(X)$を空集合に初期化する。ステップS502ではtを１に設定する。ステップS503では、図６を用いて後述するPTRの更新を行う。ステップS504ではID番号$I_m(PTR)$のプロトタイプ$Y_{Im(PTR)}$とΔ_sを計算する：

$s=I_m(PTR)$

$\Delta_s=2|\rho(Z,Y_i)|$

ただし上式の右辺第１項は前処理で計算されているので、記憶装置からの読み出しだけで良い。

【００４１】ステップS505では近傍集合$N_{t-1}(X)$にID番号とΔ_sの値とを追加する：

$N_t(X)=N_{t-1}(X)+\{(s,\Delta_s)\}$

ステップS506でtをインクレメントし、kを超えればステップS507へ進み、そうでなければステップS503へ戻る。

【００４２】ステップS507では近傍集合のなかでΔ_sの最小値をξ_{t-1}、またそれに対応するID番号sをτ_{t-1}として記憶する。以上でステップS304の近傍集合の初期化を終了する。

【００４３】ステップS305ではtをk+1に設定する。ステップS306ではPTRの更新を実行する。この処理を図６、７を用いて説明する。

【００４４】ステップS601ではBND^-_Lが０かつPTR^-_Lが０かどうかを検査し、そうであればステップS602へ、そ

うでなければステップS603へ進む。ステップS602では以下の処理を実行する：

$BND^{-}_{L}=1, Dx^{-}_{L}=\infty$

ステップS603ではBND^{-}_{H}が０かつPTR^{-}_{H}がPTR^{*}_{H}と等しいかどうかを検査し、そうであればステップS604へ、そうでなければステップS605へ進む。ステップS604では以下の処理を実行する：

$BND^{-}_{H}=1, Dx^{-}_{H}=\infty$

ステップS605ではBND^{*}_{L}が０かつPTR^{*}_{L}がPTR-Hより小さいどうかを検査し、そうであればステップS606へ、そうでなければステップS607へ進む。ステップS606では以下の処理を実行する：

$BND^{-}_{H}=1, Dx^{-}_{H}=\infty$

$BND^{*}_{L}=1, Dx^{*}_{L}=\infty$

ステップS607ではBND^{*}_{H}が０かつPTR^{*}_{H}がN2以上かどうかを検査し、そうであればステップS608へ、そうでなければステップS609へ進む。ステップS608では以下の処理を実行する：

$BND^{*}_{H}=1, Dx^{*}_{H}=\infty$

ステップS609ではBND^{-}_{L}とBND^{-}_{H}及びBND^{*}_{L}とBND^{*}_{H}の４個の数の積が１かどうかを検査し、そうであれば検索処理を終了し、そうでなければステップS610へ進む。

【００４５】ステップS610では$BND^{-}_{L}+CAL^{-}_{L}$が１かどうかを検査し、そうであればステップS611へ、そうでなければステップS612へ進む。ステップS611では以下の処理を実行する：

$Dx^{-}_{L}=|V_{m}(PTR^{-}_{L})-Z(m)|$

$CAL^{-}_{L}=1$

ステップS612では$BND^{-}_{H}+CAL^{-}_{H}$が１かどうかを検査し、そうであればステップS613へ、そうでなければステップS614へ進む。ステップS613では以下の処理を実行する：

$Dx^{-}_{H}=|V_{m}(PTR^{-}_{H})-Z(m)|$

$CAL^{-}_{H}=1$

ステップS614では$BND^{*}_{L}+CAL^{*}_{L}$が１かどうかを検査し、そうであればステップS615へ、そうでなければステップS616へ進む。ステップS615では以下の処理を実行する：

$Dx^{*}_{L}=z V_{m}(PTR^{*}_{L})-Z(m)|$

$CAL^{*}_{L}=1$

ステップS616では$BND^{*}_{H}+CAL^{*}_{H}$が１かどうかを検査し、そうであればステップS617へ、そうでなければステップS618へ進む。ステップS617では以下の処理を実行する：

$Dx^{*}_{H}=|V_{m}(PTR^{*}_{H})-Z(m)|$

$CAL^{*}_{H}=1$

ステップS618ではDx^{-}_{L}がDx^{-}_{H}、Dx^{*}_{L}、Dx^{*}_{H}のいずれよりも小さければステップS619へ、そうでなければステップS620へ進む。

【００４６】ステップS619では以下の処理を実行し、ステップS306のPTRの更新を終了する：

$Dx=Dx^{-}_{L}, PTR=PTR^{-}_{L}, CAL^{-}_{L}=0$

ステップS620ではDx^{-}_{H}がDx^{-}_{L}、Dx^{*}_{L}、Dx^{*}_{H}のいずれよりも小さければステップS621へ、そうでなければステップS622へ進む。

【００４７】ステップS621では以下の処理を実行し、ステップS306のPTRの更新を終了する：

$Dx=Dx^{-}_{H}, PTR=PTR^{-}_{H}, CAL^{-}_{H}=0$

ステップS622ではDx^{*}_{L}がDx^{-}_{L}、Dx^{-}_{H}、Dx^{*}_{H}のいずれよりも小さければステップS623へ、そうでなければステップS624へ進む。

【００４８】ステップS623では以下の処理を実行し、ステップS306のPTRの更新を終了する：

$Dx=Dx^{*}_{L}, PTR=PTR^{*}_{L}, CAL^{*}_{L}=0$

ステップS624では以下の処理を実行し、ステップS306のPTRの更新を終了する：

$Dx=Dx^{*}_{H}, PTR=PTR^{*}_{H}, CAL^{*}_{H}=0$

PTRの更新処理では、PTRと関連する変数の値を変更し、条件が満足されれば、図３に示した検索処理そのものを終了する。

【００４９】ステップS307では、次式でρ^{+}及びρ^{-}を計算する：

$\rho^{+}=\|Y_{s}\|-\|Z\|$

$\rho^{-}=\|Y_{s}\|-\|Z\|$

ステップS308では、図８につき後述する判定処理によって（５）式を評価する。

【００５０】ステップS309では、ステップS308で行った処理の戻り値がFALSEかどうかを判定し、FALSEならステップS310へ、そうでなければステップS312へ進む。

【００５１】ステップS310では、近傍集合からξ_{t-1}に対応する要素を削除し、現在処理中のプロトタイプを追加する：

$N_{t}(X)\leftarrow N_{t-1}(X)\{(\tau_{t-1},\xi_{t-1})\}+\{(I_{m}(PTR),Dx)\}$

ステップS311では、$N_{t}(X)$の要素のうちのDxの最小値をξ_{t}に、そのID番号をτ_{t}に格納し、ステップS313へ進む。

【００５２】ステップS312では次式を実行し、ステップS313へ進む：

$\xi_{t}=\xi_{t-1}$

$\tau_{t}=\tau_{t-1}$

ステップS313ではtをインクレメントし、その結果がNを超えたら処理を終了し、そうでなければステップS306へ進む。

【００５３】ステップS308の判定処理で実行する関数計算を図８を用いて説明する。

【００５４】ステップS701ではjを1に設定する。ステップS702では以下の処理を実行する：

$\rho^{+}=\rho^{+}-(Ys(\lambda_{j})-X(\lambda_{j}))^{2}$

$\rho^{-}=\rho^{-}-(Ys(\lambda_{j})+X(\lambda_{j}))^{2}$

$Dx\leftarrow Dx-(Ys(\lambda_{j})-X(\lambda_{j}))^{2}$

ステップS703では次式を判定し、満足したら戻り値にTRUEを設定して処理を終了し、そうでなければステップS704へ進む。ステップS704では、jをインクレメントし、

その結果がベクトルの次元dを超えたら、戻り値にFALSEを設定して処理を終了し、そうでなければステップS702へ戻る。

【００５５】ステップS313で終了したときの、$N_t(X)$が検索結果として出力される。

【００５６】以上説明した実施形態による効果を計算機実験によって検証する。

【００５７】〔計算機実験〕上述した実施形態の有効性を検証するために、検索結果として要求されるプロトタイプ数k＝10個、プロトタイプ数N＝10000個に対して探索の計算機実験を行った。以下の項目が実験パラメタである：

・ベクトルの次元：d＝{10から100まで、10きざみ}

実験に使用した計算機の諸元は以下のとおりである：

・CPU:PentiumIII,(500MHz)

・メインメモリ:128MB

・OS:Linux-2.0.36

尚、プログラム言語はＣを用いた。

【００５８】〔実験手順〕

(1) 一様乱数を用いて、d次元ベクトルN個より成るプロトタイプ集合を生成する。

(2) 一様乱数を用いて、d次元 ベクトル空間の計量テンソルを１個、生成する。

(3) 一様乱数を用いて、d次元のテストベクトルを１個、生成する。

(4) 全数探索を行う。

(5) 提案アルゴリズムによる探索を行う。

【００５９】上記５つの手順を10回繰り返し、以下に示す相対CPU時間の平均値を計算した。

相対CPU時間＝(提案アルゴリズムのCPU時間)/(全数探索のCPU時間)

実験結果を図１０に示す。図中、ベクトルの次元(number of dimensions)を横軸、相対CPU時間(CPU time ratio)を縦軸、プロトタイプ数Nをパラメタとして示した。

【００６０】図よりプロトタイプ数Nに関わらず、ベクトルの次元の増加に伴って相対ＣＰＵ時間が１次のオーダーで増加していることがわかる。しかしN＝10000個のときの傾きは非常に小さい。このときの相対CPU時間の値は、10次元で2%、100次元でも18%と、高次元でも十分小さい値を示している。

【００６１】＜実施形態２＞上述した類似度計算を認識装置に適用する場合について説明する。

【００６２】図１１は、本実施形態の認識装置の構成を示す図である。データベース１０３には、d次元ベクトルで表現されたN個のプロトタイプの集合が格納されており、認識対象となる各カテゴリに対してそれぞれ複数のプロトタイプが存在する。その他は図１と同様である。

【００６３】データベース１０３に格納されるｄ次元ベクトルは、例えば顔認識の場合は、各人物（Aさん、Bさん、･･･）の複数の角度、表情等における画像に、その人物の情報を関係付けることによって、構成することが出来る。このような場合、下記に述べる処理によって、入力画像と近いｋ個の画像が選択され、それぞれに関係付けられた人物番号の頻度を集計し、頻度の最も高い人物を、画像に対応する人物であると特定することが出来る。

【００６４】また、類似度計算装置１０２の処理手順は、実施形態１と同様に図２に従う。但し、ステップS204～205では、検索処理は、検索を用いた認識処理となる。この認識処理について図１２のフローチャートを用いて説明する。

【００６５】図１２において、ステップS301～313は、図３と同一である。ステップS314では、近傍集合に属するID番号に関連付けられたクラス番号の頻度を計算する。ステップS315では、クラス番号の頻度の大きい順に、ID番号を並べ替える。これにより、テストベクトルが属する可能性が高い順にクラス候補が得られる。このうち最も頻度の大きいクラスを認識結果として出力してもよいし、頻度順に所定個あるいは全部のクラスを認識候補として出力してもよい。

【００６６】尚、本発明は、単一の機器からなる装置に適用しても、複数の機器から構成されるシステムに適用してもよい。また、上述した実施形態の機能を実現するソフトウェアのプログラムコードを記憶した記憶媒体を、装置あるいはシステムに供給し、装置あるいはシステム内のコンピュータが記憶媒体に格納されたプログラムコードを読み出して実行することによって達成してもよい。

【００６７】更に、装置あるいはシステム内のコンピュータが記憶媒体に格納されたプログラムコードを読み出して実行することによって、上述した実施形態の機能を直接実現するばかりでなく、そのプログラムコードの指示に基づいて、コンピュータ上で稼動しているＯＳなどの処理により、上述の機能を実現される場合も含まれる。

【００６８】これらの場合、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。

【００６９】

【発明の効果】以上説明したように、本発明によれば、ベクトルの内積の絶対値に基づく類似度を、高速に計算することができるので、この類似度に基づいて、与えられたベクトル表現されたデータに対して、類似データの検索や属するクラスの認識を高速に実行することができるという効果がある。

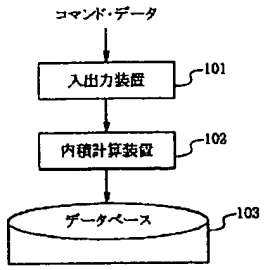
【図面の簡単な説明】

【図１】本発明に係る一実施形態のデータ検索装置の構成を示すブロック図である。

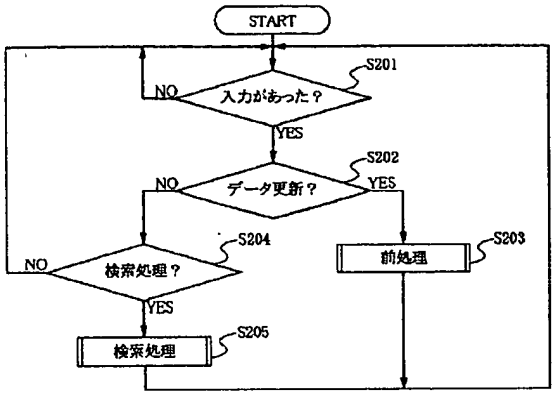
【図２】距離計算処理の手順を示すフローチャートであ

る。
【図３】検索処理の詳細手順を示すフローチャートである。
【図４】PTRの初期化手順を示すフローチャートである。
【図５】近傍集合の初期化手順を示すフローチャートである。
【図６】PTRの更新手順を示すフローチャートである。＊
＊【図７】PTRの更新手順を示すフローチャートである。
【図８】判定処理の手順を示すフローチャートである。
【図９】前処理の手順を示すフローチャートである。
【図１０】計算機実験の結果を示す図である。
【図１１】認識装置の構成を示すブロック図である。
【図１２】認識処理の詳細手順を示すフローチャートである。
【図１３】射影アルゴリズムを説明する図である。

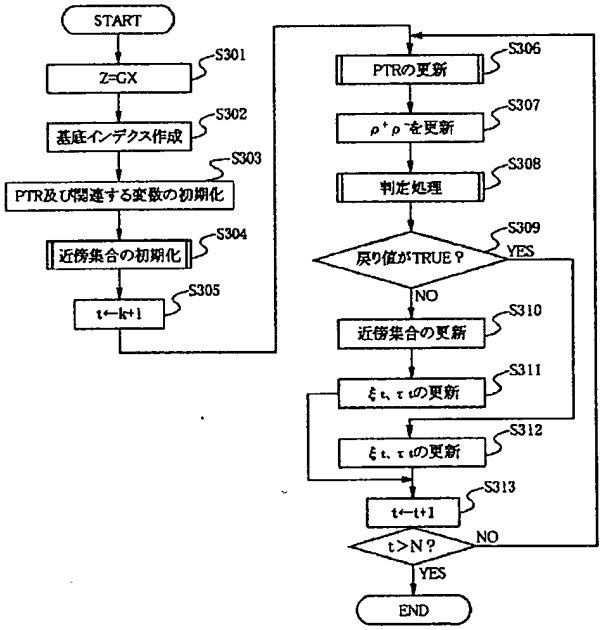
【図１】



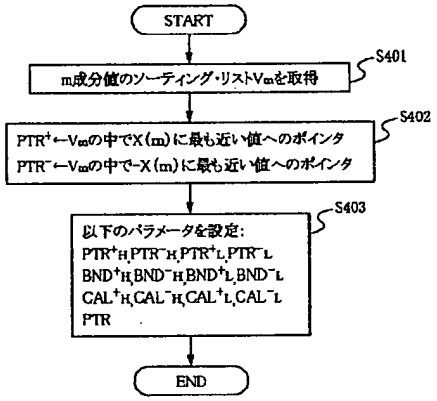
【図２】



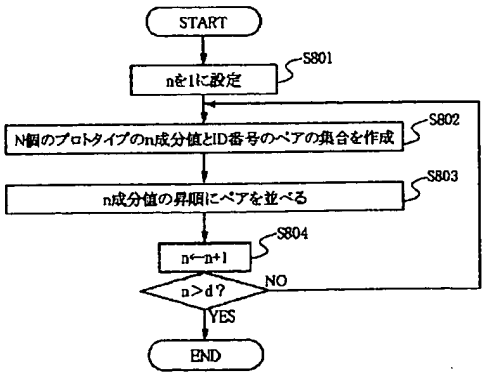
【図３】



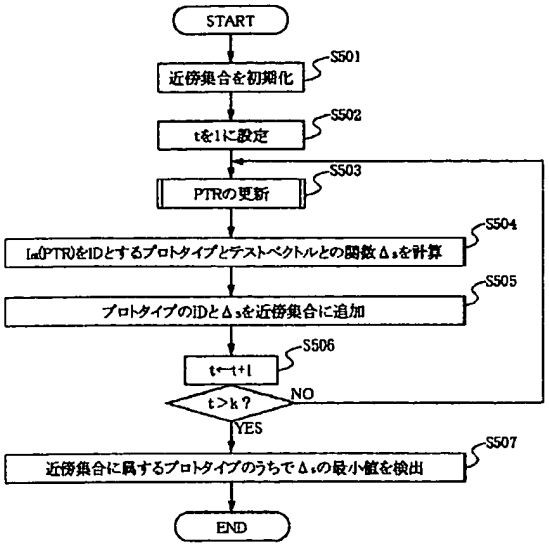
【図４】



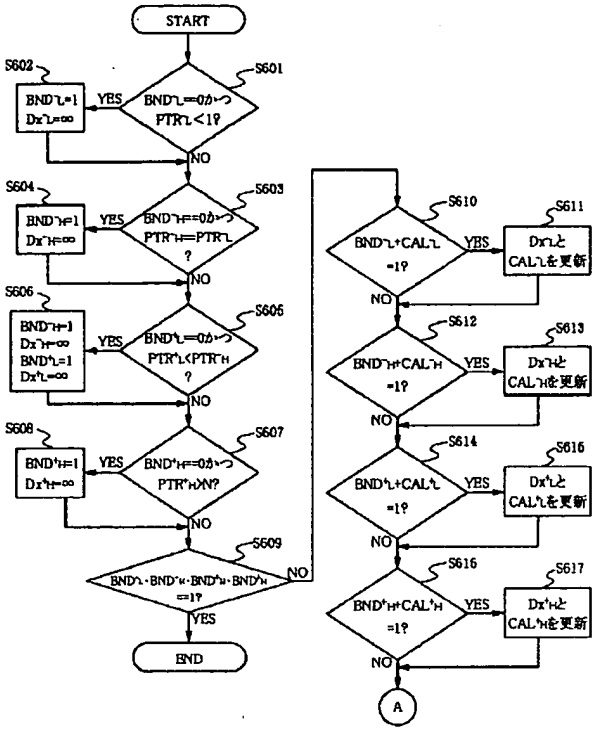
【図９】



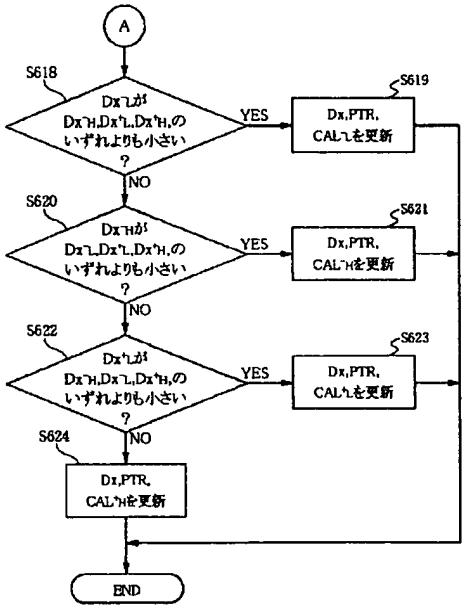
【図５】


START
近傍集合を初期化
S501
tを1に設定
S502
PTRの更新
S503
Lm(PTR)をIDとするプロトタイプとテストベクトルとの間数Δsを計算
S504
プロトタイプのIDとΔsを近傍集合に追加
S505
t←t+1
S506
t>k？
NO
YES
近傍集合に属するプロトタイプのうちでΔsの最小値を検出
S507
END

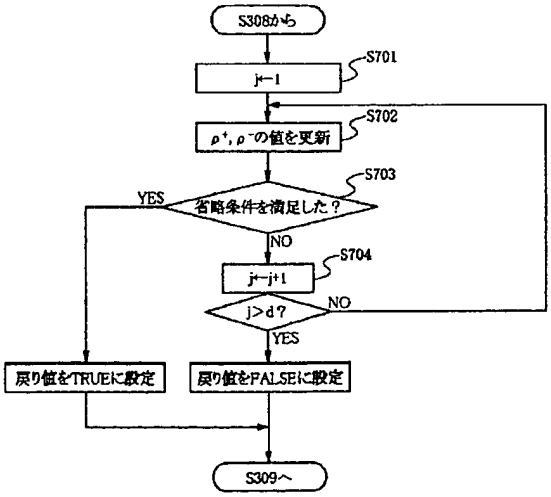
【図６】


START
S601
S602
S603
S604
S605
S606
S607
S608
S609
S610
S611
S612
S613
S614
S615
S616
S617
YES
NO
END
A

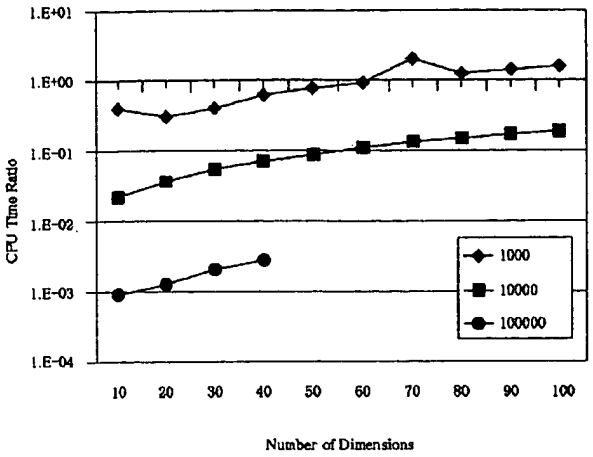
【図７】


A
S618
S619
S620
S621
S622
S623
S624
YES
NO
END

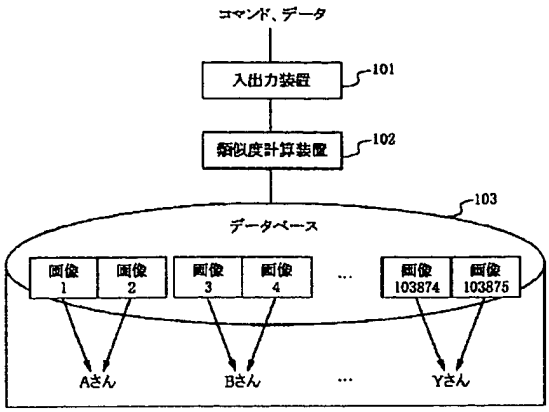
【図８】


S308から
S701
S702
S703
S704
YES
NO
戻り値をTRUEに設定
戻り値をFALSEに設定
S309へ

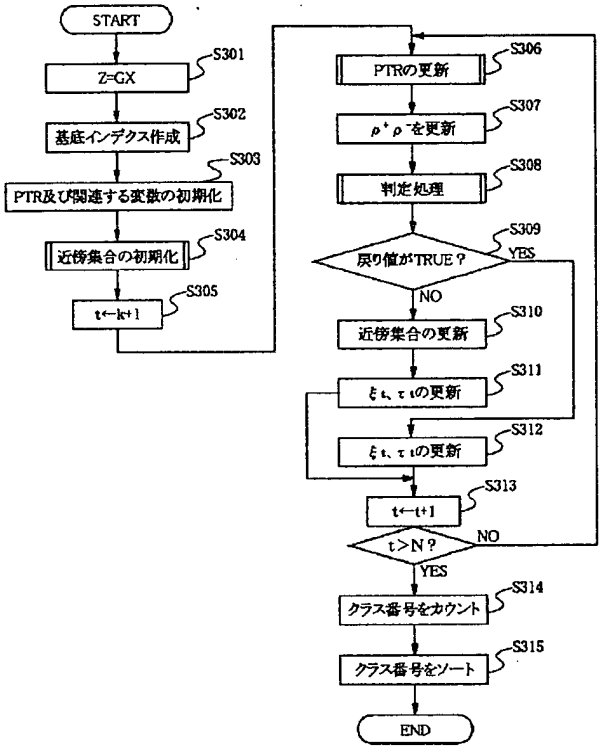
【図１０】



【図１１】



【図１２】



【図１３】